brutforce_combined_pseudo.py > ...

```python
# PSEUDO-CODE de l'Algorithme BRUTEFORCE
# 1. Définir MAX_INVEST comme 500 * 100. Cela représente le budget maximal que nous pouvons investir.
import csv
from itertools import combinations
MAX_INVEST = 500 * 100


# 2. Définir une fonction get_csv_data:
def get_csv_data():
    # - Ouvrir le fichier brutforce.csv pour la lecture.
    with open("data/brutforce.csv", newline="") as csv_file:
        csv_reader = csv.reader(csv_file, delimiter=",")
        next(csv_reader)
        # - Pour chaque ligne du fichier (en ignorant la première ligne qui est l'en-tête):
        for row in csv_reader:
            # - Extraire le nom de l'action, le prix et le bénéfice.
            stock_name = row[0]
            # - Convertir le prix et le bénéfice de l'euro en centimes.
            price_in_cents = float(row[1]) * 100
            benefit_in_cents = float(row[2]) * 100
            # - Renvoyer le nom de l'action, le prix en centimes et le bénéfice en centimes comme une ligne de données.
            yield (stock_name, int(price_in_cents), int(benefit_in_cents))


# 3. Définir une fonction generate_combinations:
def generate_combinations(stocks):
    # - Initialiser une variable profit à 0. Cela représente le profit maximal actuel.
    profit = 0
    # - Initialiser une liste best_combination vide. Cela représente la meilleure combinaison d'actions actuelle.
    best_combination = []
    # - Pour chaque sous-ensemble d'actions (obtenues en utilisant l'algorithme de combinaisons, de taille 1 à la
    #   taille totale des actions):
    for i in range(len(stocks)):
        list_combinations = combinations(stocks, i + 1)
        for combination in list_combinations:
            # - Calculer le coût total de la combinaison d'actions.
            total_cost = sum([stock[1] for stock in combination])
            # - Si le coût total est inférieur ou égal à MAX INVEST :
            if total_cost <= MAX_INVEST:
                # - Calculer le profit total de la combinaison d'actions.
                total_profit = sum([stock[2] for stock in combination])
                # - Si le profit total est supérieur au profit maximal actuel :
                if total_profit > profit:
                    # - Mettre à jour le profit maximal et la meilleure combinaison d'actions.
                    profit = total_profit
                    best_combination = combination
    return best_combination


# 4. Définir une fonction display_result :
def display_result(best_combination):
    # - Afficher le nom de chaque action, son prix et son bénéfice dans la meilleure combinaison d'actions.
    print("Liste des actions achetées :\n")
    for stock in best_combination:
        print(f"{stock[0]} {stock[1] / 100}€ {stock[2] / 100}€")
    # - Calculer et afficher la somme dépensée et le profit total.
    print(f"\nSomme dépensée : {sum([stock[1] for stock in best_combination]) / 100}€")
    print(f"Profit total : {sum([stock[2] for stock in best_combination]) / 100}€")


# 5. Si le script est exécuté en tant que programme principal :
if __name__ == "__main__":
    # - Appeler la fonction get_csv_data pour obtenir les actions.
    stocks = [stock for stock in get_csv_data()]
    # - Appeler la fonction generate_combinations pour obtenir la meilleure combinaison d'actions.
    best_combination = generate_combinations(stocks)
    # - Appeler la fonction display_result pour afficher le résultat.
    display_result(best_combination)
```